Micro Aerial Vehicle to Detect Sybil Attack in Wireless Sensor Network

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering/Information Technology

By
(Eshita Sharma (141345))

Under the supervision of

(Mr. Amol Vasudeva)

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Micro Aerial Vehicle To Detect Sybil Attack in Wireless Sensor Network"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2016 to December 2016 under the supervision of **(Supervisor name)** (Designation and Department name).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
Eshita Sharma (141345)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Supervisor Name: Mr. Amol Vasudeva
Designation: Assistant Professor
Department name: Computer Science and Engineering
Dated:

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to everyone for their exemplary guidance, monitoring and constant encouragement throughout the course of this B.Tech project.

I am highly indebted to **Mr. Amol Vasudeva**, Project Supervisor, for his guidance and constant supervision as well as for providing necessary information regarding the project. His perpetual energy, motivation, enthusiasm and immense knowledge inspired me to discipline myself in efficiently executing my multiple responsibilities simultaneously. The blessing, help, and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

WSN	Wireless Sensor Network
ID	Identity
RSSI	Received Signal Strength Indicator
GHz	Giga Hertz
MB	Mega Bytes
JVM	JAVA Virtual Machine
IDE	Integrated Development Environment
JRE	JAVA Runtime Environment
JDK	JAVA Development Kit
MANET	Mobile Ad-hoc Network
TBF	Trajectory-Based Forwarding
GEAR	Geographic and Energy-Aware Routing
GPS	Global Positioning System
Ids	Identities
MAV	Micro Aerial Vehicles
RAM	Random Access Memory

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Wireless sensor networks (WSN), are just like wireless ad hoc networks in the way that they rely on wireless connectivity and spontaneous formation of the networks in such a way that sensor data can be transported wirelessly. The WSN consists of "nodes" – varying from a few to several hundreds or even thousands, in which each node is connected to one (or sometimes even several) sensors. Wireless Sensor networks are highly indispensable for securing network protection. Highly critical attacks of a lot of kinds have been discovered and documented in the wireless sensor networks till now by many researchers. The Sybil attack is a highly massive and destructive attack against the sensor network where a lot of genuine identities with mixed forged identities are used for getting an illegal entry into a network. Here we solve this problem of detecting Sybil attack in the network using Received Signal Strength Indicator (RSSI). This solution is lightweight since we need only two nodes, that is, alongside the receiver we need the collaboration of one other node (i.e., only one message communication) for our protocol.

CHAPTER 1

INTRODUCTION

1.1 Introduction

1.1.1 Wireless Sensor Networks (WSNs)

A wireless sensor network (WSN) is a wireless network which consists of spatially distributed autonomous devices having various sensors to monitor the physical or environmental conditions. The sensor network is also connected to the Internet, an enterprise WAN or LAN, or even a specialized industrial network so that all the collected data can be transmitted to back-end systems for analysis and can be used in applications.

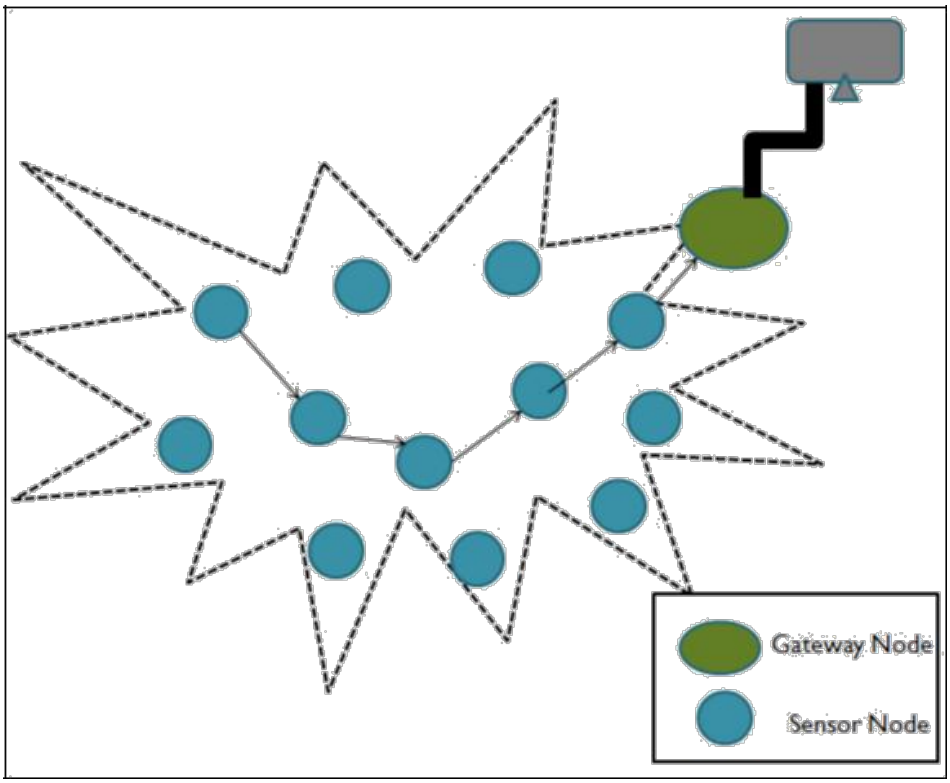

Figure 1: Wireless Sensor Network

1.1.1.1 Characteristic Table of WSNs [2]

Characteristic	Description
Low cost	There are usually hundreds or thousands of sensor nodes deployed in order to measure any physical environment. So as to reduce the overall cost of the whole network, the cost of each sensor node is kept as low as possible.
Energy efficiency	Energy in WSNs can be used for different purposes like storage, communication and computation. Sensor nodes consume more energy as compared to any other in communication. If they run out of power, they also become invalid as there is no option to recharge them.
Communication Capabilities	A WSN typically communicates by using radio waves over a wireless channel. It has the property of communicating in short range, with narrow and dynamic bandwidth. The communication channel can be both either bidirectional or unidirectional.
Security and Privacy	Every sensor node has a number of sufficient security mechanisms in order to prevent attacks, unauthorized access and unintentional damage of the information inside the sensor node.
Self-organization	The sensor nodes in the network have the capability of organizing themselves within the network as the sensor nodes are deployed in an unknown fashion and in a hostile environment. The sensor nodes work in collaboration to adjust themselves to the distributed algorithm and form the network on their own.

Table 1: WSN Characteristics

1.1.1.2 Applications of WSNs [3]

1.1.1.2.1 Area monitoring

In area monitoring, the Wireless Sensor Network is deployed over such a region where some phenomenon is to be measured and monitored. For example - A military use to detect enemy intrusion in an area.

1.1.1.2.2 Health care monitoring

There can be several medical applications of sensor networks such as wearable, implanted and environment-embedded. Wearable devices are the ones which are used on the surface of a human body. The implantable medical devices are those which are inserted inside human body. The Environment-embedded systems employ sensors which are contained in the environment. Practical applications include overall monitoring of ill patients in hospitals and at homes, body position measurement and location of persons.

1.1.1.2.3 Natural disaster prevention

WSNs can also be used to effectively prevent the consequences of natural disasters like floods by detecting them pre-hand. Deployment of wireless nodes in rivers where changes of the water levels have to be monitored is an example.

1.1.1.2.4 Data logging

For the collection of data for monitoring of environmental information also WSNs can be used. This can vary from monitoring and measuring the level of water present in overflow tanks in nuclear power plants to the monitoring of the temperature in a refrigerator. The statistical information is then be used to understand how systems are working. The "live" data feed facility which is possible in WSNs offers an advantage over the other conventional loggers.

1.1.2 Sybil attack

Sybil attack was first introduced by J. R. Douceur. According to Douceur, the Sybil attack is an attack in which a single entity, by presenting multiple identities, is capable of controlling a substantial fraction of the system.

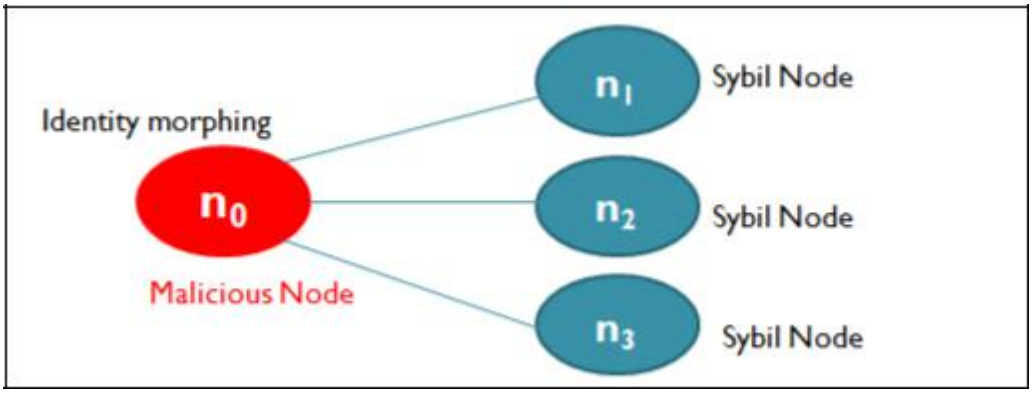

Figure 2: Sybil Attack in WSN

A Single user (that is, attacker) corrupts the system of a distributed and decentralized network. The attacker does this by creating a large number of pseudonymous or Sybil identities. These identities are then used to gain a disproportionately large influence. The node which spoofs the identities of the nodes is called a malicious node while Sybil nodes area the nodes whose identities are spoofed.

Figure 2 represents a malicious node with its three Sybil nodes. When this malicious node communicates with any legitimate node with all of its three identities, the legitimate node forms the illusion that it has communicated with three different nodes. But in reality, there exists only one physical node with multiple different Ids.

1.1.2.1 Dimensions of Sybil Attack [1]

Sybil attack can be classified into 3 dimensions, which are communication, participation and identity.

I. Communication

-Direct Communication

- In Direct Communication, legitimate nodes can communicate with the Sybil nodes directly.

-Indirect Communication

- One or more of the malicious devices claims that it is able to reach the Sybil nodes.
- Messages that are sent to a Sybil node are routed through one of these malicious nodes.
- Malicious nodes pretend that they are passing on the message to a Sybil node.

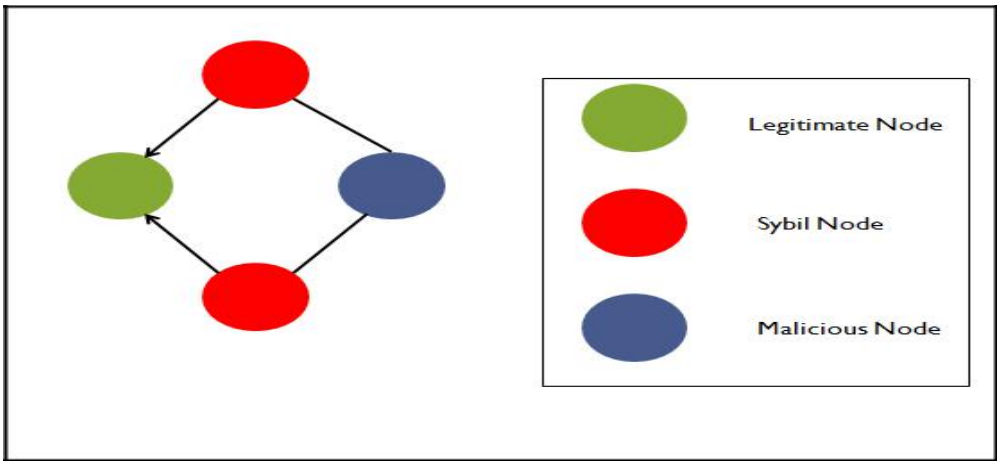

Figure 3: Direct Communication

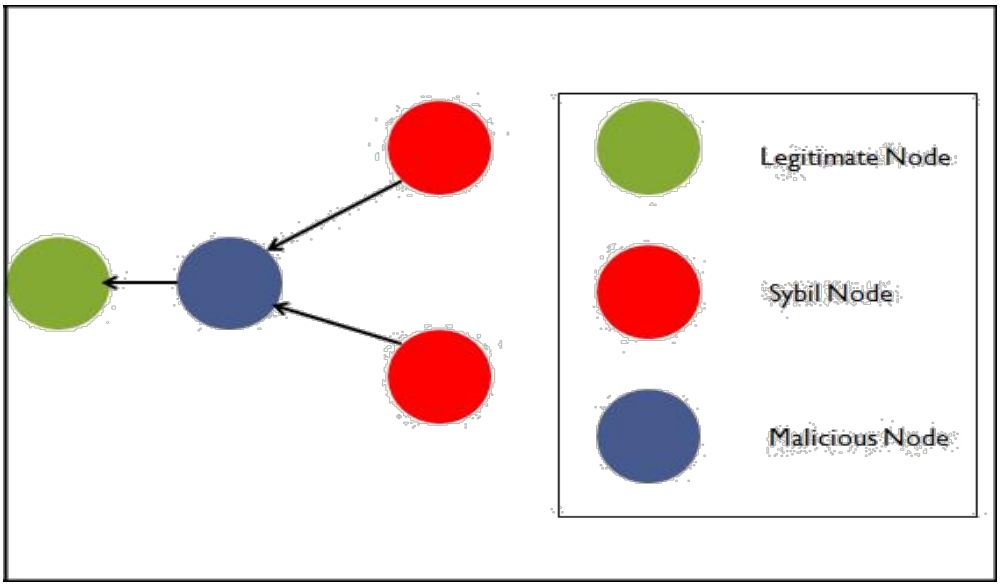

Figure 4: Indirect Communication

II. Participation

- **Simultaneous**
  - The attacker tries to have all his Sybil identities participate in the network all at once.
  - The hardware entity cycles through identities so that it appears that they are all present simultaneously.

- **Non-Simultaneous**
  - Attacker presents a large number of ID's over a period of time, while only a smaller number of identities act at any given point of time.
  - The attacker can do this by having one ID seem to leave the network, and then have another identity join in its place.

III. Identity

- **Fabricated Identity**

  - The attacker creates new identities arbitrarily.

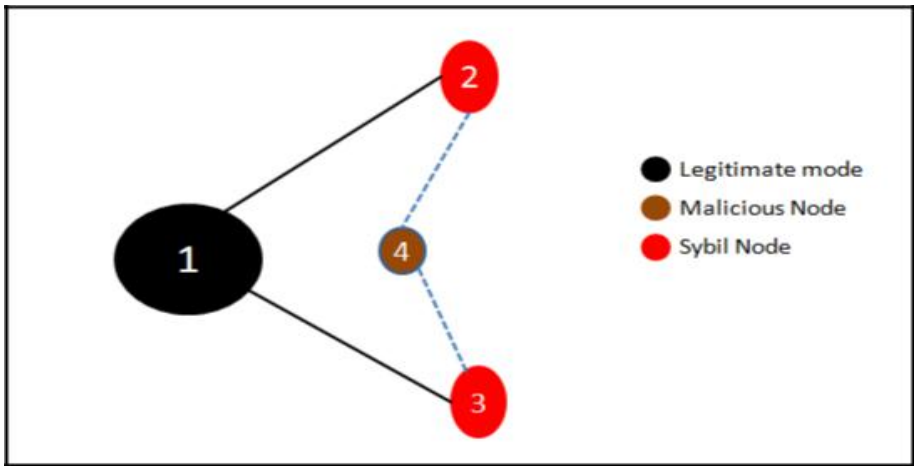

Figure 5: Fabricated Identity

- **Stolen Identity**

  - Sybil nodes are assigned legitimate identities.
  - The attack can go undetected if the attacker destroys or disables the original identity**.**

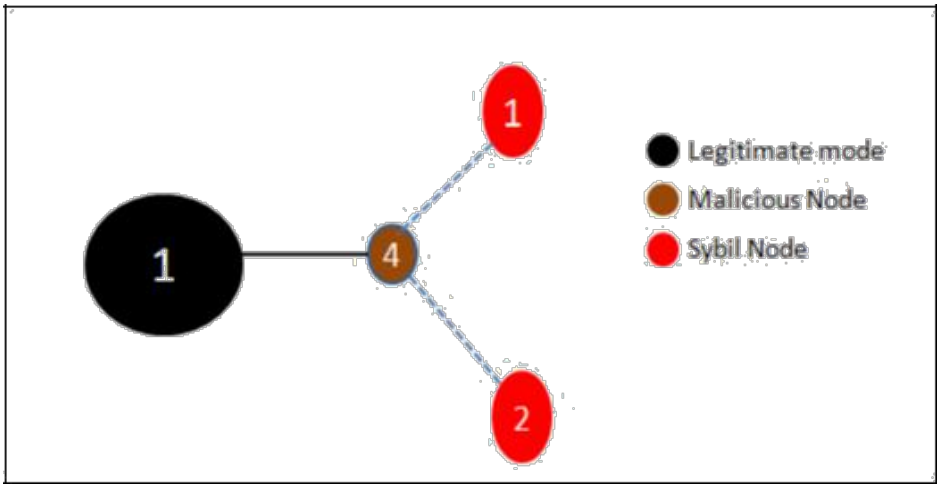

Figure 6: Stolen Identity

IV. Sybil attack effects

- Routing

Sybil attack can have a very big impact on the functionality of the network. A malicious node can involve into a multiple path, by presenting multiple identities so that it is able to disrupt the routing procedure.

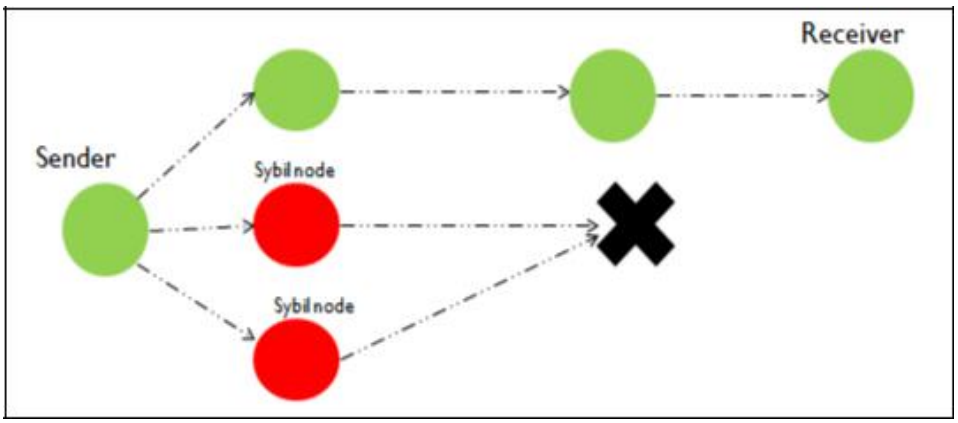

Figure 7: Sybil attack effect on Routing

- Data Aggregation

Instead of returning readings of individual sensors, efficient query protocols compute aggregates of sensor readings within the network. This is done in order to conserve energy. By using the Sybil attack, a single malicious node may be able to contribute to the aggregate multiple times. With enough Sybil nodes, an attacker is also capable of completely altering the aggregate reading.

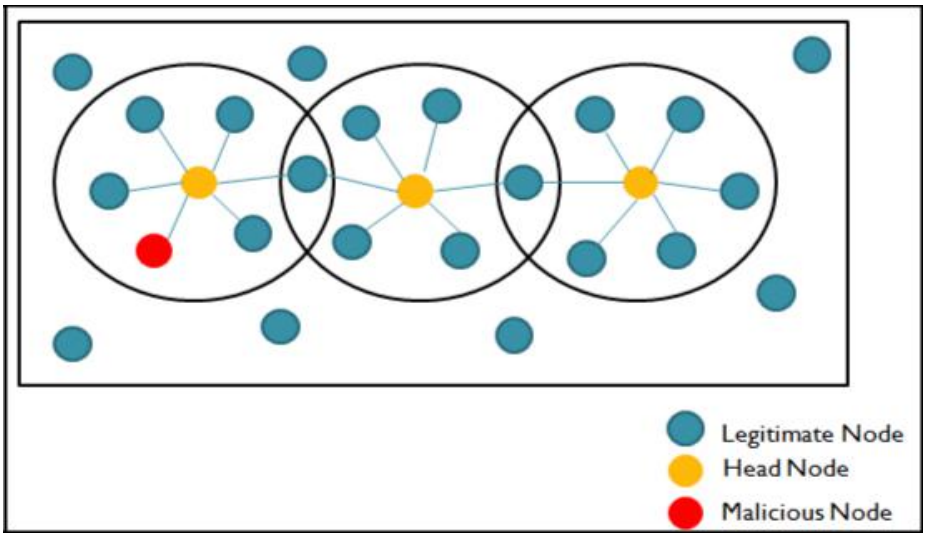

Figure 8: Sybil attack effect on Data aggregation

- **Voting**

WSNs could use voting procedure for a number of tasks. The Sybil attack can also "stuff the ballot box" in any such voting process. The attacker may be able to determine the outcome of any vote depending on the number of identities that the he or she owns.

- **Misbehavior Detection**

The network is capable of potentially detecting any particular type of misbehavior taking place. Since misbehavior detector does not take any action until it observes a lot of repeated offenses by the same node, an attacker with multiple Sybil nodes can "**spread the blame**", by not having any one single Sybil identity misbehave enough, and hence the system cannot take any action.

1.1.2 Micro Aerial Vehicles (MAVs)

A micro air vehicle (MAV) belongs to the class of miniature UAVs that usually comes with a size restriction. The MAVs can be either autonomous or can be even human controlled. The size of the modern day craft can be as small as 5 centimeters. The development in the field of MAVs is motivated by commercial, government, research, and as well as military purposes. [4] The insect-sized aircraft, which is expected soon in the future, is assumed to be the best in the field. The small sized craft allows the remote observation of hazardous and hostile environments which are generally inaccessible to ground vehicles and other human inventions.

The application areas of MAVs are varied and include fields like for hobby purposes, such aerial robotics contests and aerial photography. The use of Micro Aerial Vehicles in the WSNs is given a lot of importance since it allows easy and accurate collection of data in hostile environments. This accumulated data can then also be easily processed since the MAVs make it extremely easy to forward this data to the data centers. With the help of well established internet connectivity and location mechanism, the processing and computation of the data has become easy and efficient.

The benefits of using MAVs to detect the Sybil attack in Wireless Sensor Networks include conservation of energy, easy processing and communication of data as well as optimized utilization of all the available resources of the network.

1.2 Problem Statement

Wireless Sensor Network (WSN) has various characteristics which makes it susceptible to external attacks which can disrupt the network. Sybil attack is one of such attacks which utilizes the benefits provided by the WSN characteristics and then disrupts the wireless network.

In this project, instead of using mechanism of encryption keys (traditional method), we will use Received Signal Strength Indicator (RSSI) to detect the presence of Sybil attack in WSNs, a method which is based upon the measurement of power present in the radio signal received by the Micro Aerial Vehicles(MAVs). This requires building a system in which a network would be virtually established and the nodes will interact with each other as in case of WSNs. Further, minimum two receiver nodes, one of which will be a master node are required. Once the network is established, we will implement our algorithm to detect the Sybil attack. Detecting would require placing a malicious node which would further create the Sybil nodes by impersonating its identity.

Hence, the algorithm presents a lightweight solution to detect the Sybil attack.

1.3 Objectives

- To analyze WSN attacks and their impact on the quality of service.
- To investigate the Sybil attack and its effect on WSN.
- To propose an algorithm to detect the attack in the sensor network.
- To perform a Sybil attack in WSN and disrupt the entire network.
- To study the effects and consequences of a Sybil attack.
- To implement our algorithm using swing, applets, Servlets, etc.

1.4 Methodology

This section covers the detailed methodology which is necessary to make this project complete and efficient. In order to evaluate this project, the Systems Development Life Cycle (SDLC), consisting of three major phases namely, planning, implementation and analysis are used.

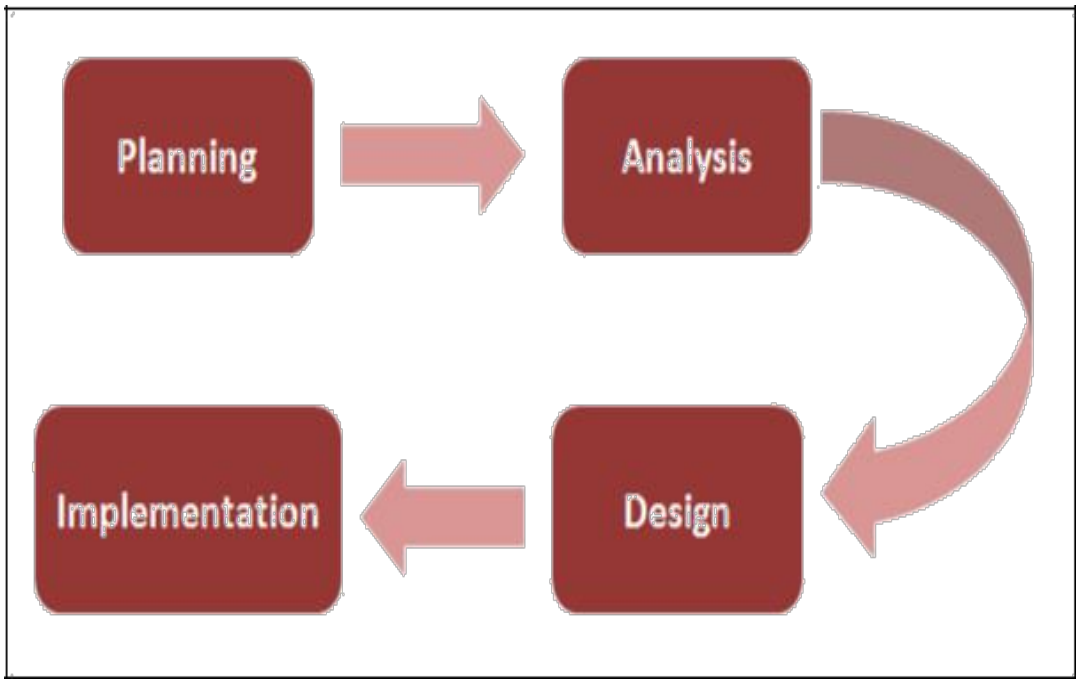

Figure 9: SDLC Flowchart

The phases are described as:

1. **Planning**
    - Data collection
    - Software and Hardware requirements

2. **Analysis**
    - Analyze the performance
    - Identify the conclusion

3. **Design**
    - Transform business requirements into a detailed system architecture which is feasible and robust.

4. **Implementation**
    - Testing point
    - Implement the project

1.5 Organization

Chapter 1 highlights and underlines the characteristics of a WSN. In this chapter various security issues arising within the WSN are discussed and then the various attacks which utilize them are discussed. The key focus however remains on how to detect Sybil attack in Wireless Sensor Network using RSSI.

The detailed literature review from the research paper, books and journals are done in **Chapter 2**. In this chapter, the extracts from assorted research papers on detection of Sybil attack using RSSI are taken and depicted.

Chapter 3 covers the system development which is the key aspect of this work. In this chapter, the proposed model, algorithm and related parameters are emphasized.

The simulation of implementation results with the relative performance analysis is shown in **Chapter 4**. In this chapter, the simulation results and screenshots are revealed to depict and defend the proposed work.

Chapter 5 ends with the detailed conclusion and scope of the future work which guides the upcoming students and research scholars to enhance the current work with higher efficiency and effectiveness.

CHAPTER 2

LITERATURE SURVEY

To completely justify and solve the problem definition, a number of research papers, magazines, journals and online links were investigated in detail. In this chapter, details of research papers and journals are specified from where we have analyzed the content and formulated the problem

A number of research scholars and scientists have written a number of research papers and found excellent results. This section underlines all those research papers and their extracts.

2.1 An RSSI-based Scheme for Sybil Attack Detection in Wireless Sensor Networks by Murat Demirbas and Youngwhan Song [5]

In this research paper, we learnt about Sybil attack and the concept of Sybil nodes. It also explained about the RSSI solution to Sybil attack, all of which are described below.

Sybil Attack

Sybil attack is an attack wherein an attacker devises its multiple identities and then uses them to disturb and disrupt the normal functioning of the network. Communication medium used in Wireless Sensor Network (WSN) is broadcast. The frequency shared among the various nodes is also same; therefore it becomes easy to perform Sybil attacks in such networks. By broadcasting different messages with multiple identifications, a Sybil node can manipulate the vote on group based decisions and also affect severely the network middleware services.

RSSI Based Solution Sybil Attack Detection [5]

A received signal strength indicator (RSSI) based solution for detecting the Sybil attack is desirable since it does not require piggybacking of keys to messages and shared keys. We have a receiver which detects the attack by associating the RSSI of the message with the sender-id also included in the message, and later when another different message with same RSSI but with another sender-id is received, the receiver complains of a Sybil attack. But since RSSI has time-varying nature, this method fails. Moreover, since the transmission power in WSN is fairly easy to change, a Sybil node can easily send messages with different IDs using varied transmission power to trick the receiver. Since RSSI is a function of transmission power, different RSSI readings will be obtained due to different transmission powers.

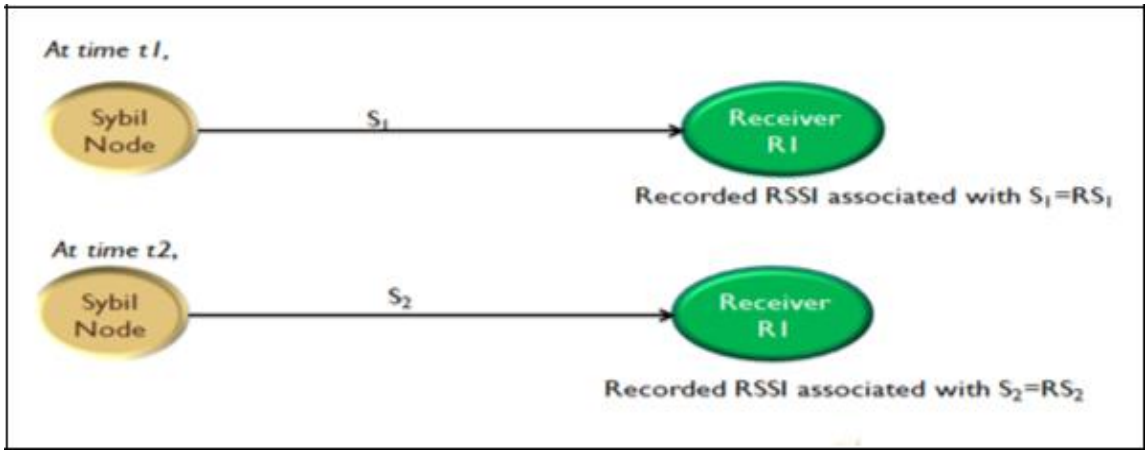

Figure 10: Sybil attack detection (If $RS_1 = RS_2$, Sybil attack detected)

The solution mentioned in the paper is lightweight since along with the receiver, we require the coactions of only one other node. The above problems (of time varying and unreliable nature of RSSI) can be removed easily by using the ratio of RSSIs from multiple receivers.

If only one receiver is used, there is a lot of variation in RSSI values, however by using multiple receivers and the ratio of RSSIs, the time variance of RSSI is overpowered and the standard deviation is very small.

Suppose node *j* receives radio signal from node 1, then the RSSI is $R_j = P_0 K / d_j^{\alpha}$ where

P_0 represents transmitter power,

R_j is RSSI,

K is constant,

d_j is Euclidean distance,

α is distance-power gradient.

The RSSI ratio of node *i* to *j* is

$$R_i/R_j = (\frac{P_0 \cdot K}{d_i^{\alpha}})/(\frac{P_0 \cdot K}{d_j^{\alpha}}) = (\frac{d_i}{d_j})^{\alpha}$$

In this above equation, since the values of P_0 cancel out in the ratio of RSSIs, this technique remains unaffected by the changes to the transmission power P_0.

When a message is received, the four installed detector nodes first compute the location of the sender and associate this location with the sender id which is also included in the message. After sometime, when another message is received and the location of the sender is computed to be the same as the previous time, a Sybil attack is detected.

2.2 The Sybil Attack in Sensor Networks: Analysis & Defenses by James Newsome [4]

2.2.1 Introduction

A very crucial and complicated issue in sensor networks is security because of the broadcast nature possessed by the wireless communication. In this paper, the Sybil attack, which is a particularly harmful attack in sensor networks is discussed in great detail. In a Sybil attack, a single malicious node behaves as if it were a larger number of nodes, for example by imitating other nodes or simply just by claiming false identities. In the worst case, an attacker by using only one physical device, may generate an arbitrary number of additional node identities.

2.2.2 Sybil attack taxonomy

2.2.2.1 Dimension 1: Direct versus Indirect Communication

Direct Communication: In this method, a direct communication between the Sybil nodes and the legitimate nodes is established. When a radio message is sent from a legitimate node to a Sybil node, one of the malicious devices secretly listens to the message. Similarly, messages which are sent from Sybil nodes are actually simply sent from one of the malicious devices.

Indirect Communication: In this type of attack, there is no direct established communication between the legitimate nodes and the Sybil nodes. Instead, one or more of the various malicious devices claims that it is able to reach the Sybil nodes. Messages sent to a Sybil node are then routed through one of these malicious nodes, which just pretend to pass on the message to a Sybil node.

2.2.2.2 Dimension 2: Fabricated versus Stolen Identities

Fabricated Identity: In some cases, the attacker can simply create new arbitrary Sybil identities. For example, if each node is identified using a 32-bit integer, the attacker can simply allot each Sybil node any random 32-bit value.

Stolen Identity: If there already exists a mechanism to identify legitimate nodes in the network, an attacker just cannot fabricate new identities. In this case, the attacker has to assign other legitimate identities to Sybil nodes. If the impersonated nodes are temporarily disabled or destroyed, this identity theft may go undetected.

2.2.2.3 Dimension 3: Simultaneity

Simultaneous: The attacker may try to have all his Sybil identities participate in the network at once, that is, at the same time. While a particular hardware entity can act only as one identity at any point of time, it can cycle through these identities to make sure that it appears that they are all present simultaneously.

Non simultaneous: In contrast to the simultaneous attack, the attacker might actually present a large number of identities over a period of time, while however allowing only a smaller number of identities to act at any given time. The attacker is able to do this by having one identity seem to leave the network, and then have another identity join the network in its place.

2.2.3 Attacks

2.2.3.1 Known attacks

Distributed Storage- Sybil nodes having fake multiple identities collect maximum of data from the network which is assumed to be with multiple legitimate nodes but is in reality with the single Sybil node.

Routing- Multiple routing paths are disrupted by creating multiple fake identities as it pretends to be in many places so it can collect as many as possible of the transmitted messages.

2.2.3.2 New Attacks

Data Aggregation- A Lot number of false data sets are created with the multiple identities of the Sybil nodes which are assumed to be coming from many legitimate nodes.

Voting- Sybil node acts like a selfish node and creates multiple number of voters to increase its votes during the election of cluster head. Once it becomes the cluster head, then all messages start to pass through it and it gets the entire data about what is happening in the simulation environment.

Resource Allocation- Sybil Nodes act like greedy nodes in the resource allocation and get more shares of the overall resources present in the network. This leads to depletion of resources.

2.3 Routing Protocols in Wireless Sensor Networks by Shio Kumar Singh, M P Singh and D K Singh [6]

This paper explains the characteristics of the network, the design of the network as well as the challenges and the routing protocols in detail.

The routing protocols for wireless sensor networks are responsible for the definition as well as the maintenance of the routes in the network. Along with this, they also have to ensure reliable multi-hop communication under these conditions.

2.3.1 Network Characteristics

In comparison to the traditional wireless communication networks like cellular systems and mobile ad hoc networks (MANETs), wireless sensor networks have the following unique constraints and characteristics:

- **Battery-powered sensor nodes**: Sensor nodes are usually operated by battery and are deployed in a hostile environment where it is very difficult to change or recharge the existing batteries.

- **Dense sensor node deployment**: Sensor nodes are often densely deployed and the number of deployed sensors can be several orders of magnitude higher than that in a MANET.

- **Severe energy, storage constraints and computation**: Sensors nodes mostly have highly limited energy, limited computation, and limited storage capabilities.

- **Self-configurable**: Sensor nodes are often randomly deployed and hence autonomously have to configure themselves to form a communication network.

- **Many-to-one traffic pattern**: In most sensor network applications, a many-to-one traffic pattern is exhibited since the data sensed by the deployed sensor nodes flows from multiple source sensor nodes to a particular sink node.

- **Frequent topology change**: Network topology changes frequently in a WSN due to the, damage, addition, node failures, energy depletion, or channel fading.

2.3.2 Network Design Challenges and Routing Issues

The routing protocol designs for WSNs are challenging because of several constraints offered by the network. The following main aspects are involved in the design challenges in sensor networks:

- **Limited energy capacity:** Since the sensor nodes are battery operated, they have very limited energy capacity. Thus, energy poses a great challenge for network designers in tough environments.

- **Sensor locations**: Another major challenge regarding the design of routing protocols is the management of the locations of sensors. It is assumed by most of the proposed protocols that the sensors are either equipped with global positioning system (GPS) receivers or a localization technique is used by them to learn about their locations.

- **Limited hardware resources:** In addition to limited energy capacity, sensor nodes also have limited processing and storage capacities, and thus are only capable of performing few, limited computational functionalities.

- **Massive and random node deployment**: The Sensor node deployment in WSNs is application dependent and can thus be either random or manual which ultimately affects the performance of the used routing protocol. In most applications, sensor nodes are scattered randomly in the intended area or dropped massively over an inaccessible or hostile region randomly. If the overall resultant distribution of nodes is not uniform, optimal clustering of nodes becomes necessary to enable connectivity and allow energy efficient operation in the network.

- **Network characteristics and unreliable environment**: A sensor network is usually operated in an unreliable and dynamic environment. The network topology, which is defined by the sensors and the links for communication between these sensors, changes frequently due to deletion, node failures, sensor addition, damages, or even energy depletion.

- **Data Aggregation:** Since sensor nodes can generate significant amount of redundant data, similar packets coming from multiple nodes are aggregated so that the number of overall transmissions is reduced. This data aggregation technique is used to achieve energy efficiency and optimized data transfer in a lot of routing protocols.

2.3.3 Routing Protocols in WSN

2.3.3.1 Geographic Adaptive Fidelity (GAF)

GAF is an energy-aware routing protocol which is primarily projected for MANETs, but can also be used in WSNs because it supports energy conservation. The design of GAF is based on inspiration from an energy model that considers energy consumption arising due to the transmission and reception of packets and as well as idle (or listening) time, that is, when the radio of the sensor is on to detects the incoming packets.

In GAF, the sensor field is divided into various grid squares and every sensor uses its location information, which is provided either by GPS or other location systems, to associate itself to a particular grid in which it is currently residing. This kind of association is misused by GAF to identify those sensors that are equivalent from the view of packet forwarding.

2.3.3.2 Trajectory-Based Forwarding (TBF)

TBF is a routing protocol which requires presence of a sufficiently dense network and a coordinate system, like for example, a GPS, so that the sensors are able to position themselves and estimate distance to their respective neighbors. The trajectory in a packet is specified by the source, but the explicit path on a hop-by-hop basis is not indicated. By using the location information from its neighbors, a forwarding sensor makes a greedy decision about determining the next hop which will be the closest to the trajectory fixed by the source sensor.

2.3.3.3 Geographic and Energy-Aware Routing (GEAR)

GEAR is an energy-efficient routing protocol mainly proposed for routing queries to target regions in a sensor field. In GEAR, the sensors already have localization hardware equipped, such as, a GPS unit or a localization system so that their current positions are known. Furthermore, the sensors are also aware of their residual energy and the locations and residual energy of each of their respective neighbors. An energy aware heuristics is used by GEAR based on geographical information to select sensors to route a packet toward its destination region.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 Algorithm for Sybil attack detection using RSSI [5]

Assumption

- Network is static where all nodes are immobile after initial deployment is done.
- Initial set of nodes that are trustworthy (non-Sybil).
- Later, as a part of re-populating the network, new nodes is introduced some of which can be Sybil as well.

Steps

- At t_1, each monitoring node records the RSSI values and forged ID S_1. $^{S1}R_{D1}$ is the recorded RSSI by Master Node D_1 from the Sybil node with ID S_1.
- Similarly $^{S1}R_{D2}$, $^{S1}R_{D3}$, $^{S1}R_{D4}$ are the RSSI calculated by D_2, D_3 and D_4.
- D_1, D_2 and D_3 send these calculated RSSI values from S_1 to D_1.
- Thus D_1 accumulates all the messages from the monitors and computes the ratio as:

$$\frac{^{SI}R_{D1}}{^{SI}R_{D2}}, \frac{^{SI}R_{D1}}{^{SI}R_{D3}}, \frac{^{SI}R_{D1}}{^{SI}R_{D4}}$$

- Similarly at t_2, the Sybil node uses a different ID S_2 and D_1 computes each ratio as

$$\frac{{}^{S2}R_{DI}}{{}^{S2}R_{D2}} \quad \frac{{}^{S2}R_{DI}}{{}^{S2}R_{D3}} \quad \frac{{}^{S2}R_{DI}}{{}^{S2}R_{D4}}$$

- Now if,

$$\left(\frac{{}^{S1}R_{D1}}{{}^{S1}R_{D2}} = \frac{{}^{S2}R_{D1}}{{}^{S2}R_{D2}}\right), \left(\frac{{}^{S1}R_{D1}}{{}^{S1}R_{D3}} = \frac{{}^{S2}R_{D1}}{{}^{S2}R_{D3}}\right) \text{and} \left(\frac{{}^{S1}R_{D1}}{{}^{S1}R_{D4}} = \frac{{}^{S2}R_{D1}}{{}^{S2}R_{D4}}\right)$$

Then D_1 detects a Sybil attack, since if the RSSI ratios are same it means the location is in fact same for alleged multiple ID's.

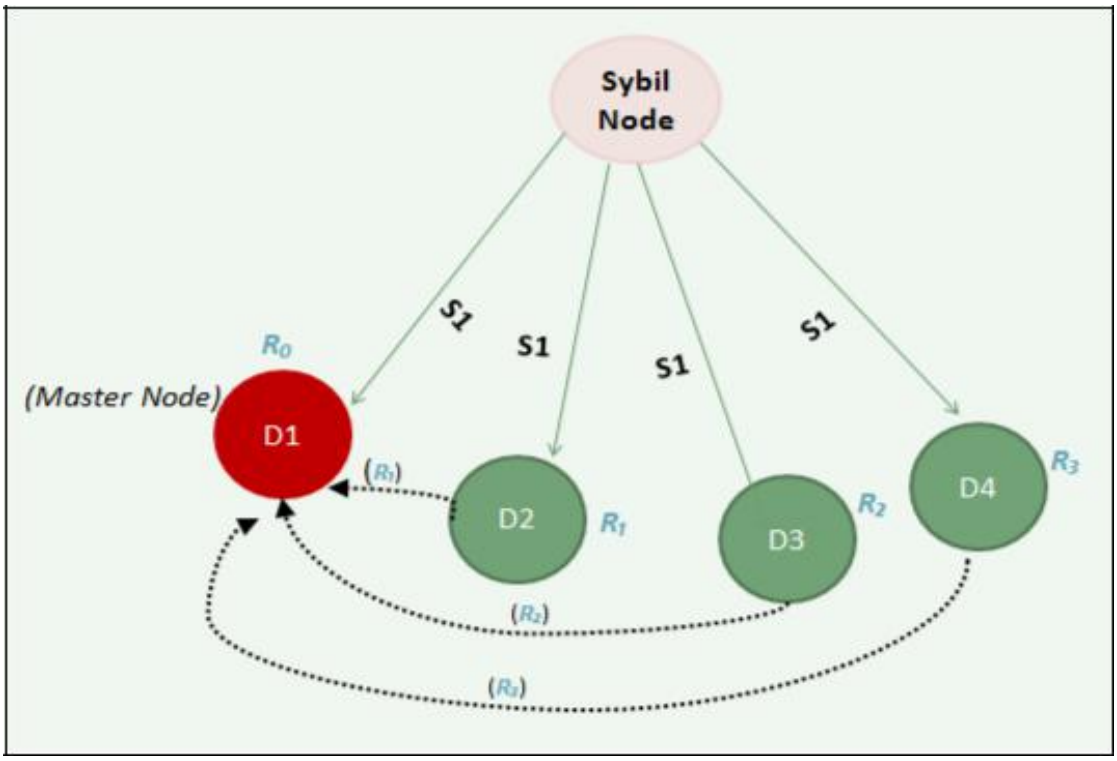

Figure 11: RSSI Solution to Sybil Attack

3.2 Problems with the existing system

The problem with existing system is that the above stated solution only works with the static system (nodes are static) and can’t be implemented with mobile wireless sensor network.

The other problem is that the battery of malicious node will get drained after the particular period of time, due to regular communication with the target node. Thus the malicious node is required to be equipped with more resources in terms of memory, battery and processing power.

3.3 Software development model

Prototype model is used for software development. As our application is interactive, prototype model is best suited for such applications.

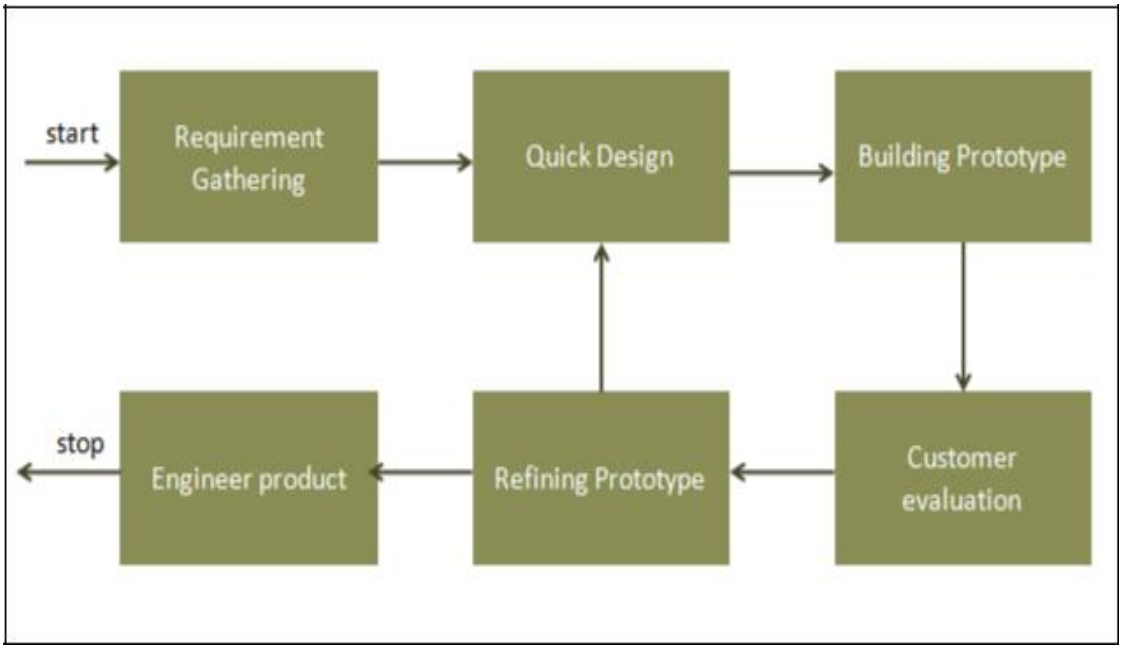

Figure 12: Prototype Model

3.4 Hardware Requirements

The minimum requirements needed to perform operations are

- Intel Pentium Processor at 2 GHz or Higher
- RAM 256MB or more
- Hard disk capacity 10 GB or more

3.5 Software Requirements

The software required to perform the implementations are

- Windows or Linux Operating System (Ubuntu, Fedora)
- JDK 8 and above
- Eclipse/ NetBeans IDE
- Dia – The Diagram Editor

CHAPTER 4

PERFORMANCE ANALYSIS

The practical implementation of the algorithm has been discussed below.

We have taken fifteen mobile nodes in the algorithm. All the nodes have been randomly deployed and are moving in random directions. These fifteen nodes include one malicious node and three observer nodes.

The specifications are discussed as shown:

- The observer nodes have identities 0, 5 and 10, that is, Node with Id 0, Node with Id 5, Node with Id 10 are all observer nodes, the micro aerial vehicles. The other nodes in the network are not aware that these nodes are the observer nodes which have been deployed for observation purposes.

- There exists one malicious device. We have taken the node with Identity number 12 as that device. The malicious device has 2 Sybil identities, that is, nodes with identity numbers 13 and 14.

The algorithm first computes the transmission power for all the fifteen nodes. The transmission range has been set as 200 meters and is same for all the nodes.

Then the RSSI values are calculated for all the nodes. Then the ratios are computed. The process is repeated for a fixed number of iterations.

The data for the first iteration is shown as below.

Transmission power of 0	3.2102955
Transmission power of 1	3.2102955
Transmission power of 2	3.2102955
Transmission power of 3	3.2102955
Transmission power of 4	3.2102955
Transmission power of 5	3.2102955
Transmission power of 6	3.2102955
Transmission power of 7	3.2102955
Transmission power of 8	3.2102955
Transmission power of 9	3.2102955
Transmission power of 10	3.2102955
Transmission power of 11	3.2102955
Transmission power of 12	3.2102955
Transmission power of 13	3.2102955
Transmission power of 14	3.2102955

Table 2: Transmission power of nodes

The RSSI values for the first iteration are calculated as below:

```
RSSI of 0 with 0 is 0.0
 RSSI of 0 with 1 is 1.0571615605436557
 RSSI of 0 with 2 is 0.9752127491054292
 RSSI of 0 with 3 is 1.0964878967077873
 RSSI of 0 with 4 is 0.9771486670014439
 RSSI of 0 with 5 is 0.9604944675728343
 RSSI of 0 with 6 is 1.0840965069515947
 RSSI of 0 with 7 is 0.9949400822905202
 RSSI of 0 with 8 is 0.9528996181221296
```

```
RSSI of 0 with 9 is 1.0080394005708233
RSSI of 0 with 10 is 0.9405815168628735
RSSI of 0 with 11 is 1.0330692709625744
RSSI of 0 with 12 is 0.9450812988107802
RSSI of 0 with 13 is 0.9450812988107802
RSSI of 0 with 14 is 0.9450812988107802
RSSI of 5 with 0 is 0.9604944675728343
RSSI of 5 with 1 is 0.7590898727450529
RSSI of 5 with 2 is 0.9667802462468226
RSSI of 5 with 3 is 1.1641767296166463
RSSI of 5 with 4 is 0.9969911825130237
RSSI of 5 with 5 is 0.0
RSSI of 5 with 6 is 1.0394231976582136
RSSI of 5 with 7 is 1.520847806751962
RSSI of 5 with 8 is 1.0316572325510744
RSSI of 5 with 9 is 1.7468739122331889
RSSI of 5 with 10 is 0.9815911620803399
RSSI of 5 with 11 is 0.6704131046142332
RSSI of 5 with 12 is 0.9597559835903972
RSSI of 5 with 13 is 0.9597559835903972
RSSI of 5 with 14 is 0.9597559835903972
RSSI of 10 with 0 is 0.9405815168628735
RSSI of 10 with 1 is 1.040371742798496
RSSI of 10 with 2 is 0.9662890183445829
RSSI of 10 with 3 is 1.1016838464831773
RSSI of 10 with 4 is 1.16720755521899
RSSI of 10 with 5 is 0.9815911620803399
RSSI of 10 with 6 is 1.017284091816972
RSSI of 10 with 7 is 1.1774131157928232
RSSI of 10 with 8 is 0.9169262111791366
RSSI of 10 with 9 is 1.1166334252897645
RSSI of 10 with 10 is 0.0
RSSI of 10 with 11 is 0.8245413165806019
RSSI of 10 with 12 is 0.9537887006006092
RSSI of 10 with 13 is 0.9537887006006092
RSSI of 10 with 14 is 0.9537887006006092
```

This detects that 12 is the malicious node.

Similarly the whole output has been shown as:

Iteration 2:

Transmission power of 0	3.2102955
Transmission power of 1	3.2102955
Transmission power of 2	3.2102955
Transmission power of 3	3.2102955
Transmission power of 4	3.2102955
Transmission power of 5	3.2102955
Transmission power of 6	3.2102955
Transmission power of 7	3.2102955
Transmission power of 8	3.2102955
Transmission power of 9	3.2102955
Transmission power of 10	3.2102955
Transmission power of 11	3.2102955
Transmission power of 12	3.2102955
Transmission power of 13	3.2102955
Transmission power of 14	3.2102955

Column1	Column2	Column3
RSSI of 0 with 0 is 0.0		
RSSI of 0 with 1 is 0.6186734084455815		
RSSI of 0 with 2 is 0.7476771076761952		
RSSI of 0 with 3 is 1.0448443375753		
RSSI of 0 with 4 is 0.9331336791721855		
RSSI of 0 with 5 is 0.8973189462481387		
RSSI of 0 with 6 is 0.7803861097542261		
RSSI of 0 with 7 is 0.8708926072042682		

RSSI of 0 with 8 is 1.0399900753734934
RSSI of 0 with 9 is 0.7781448646924207
RSSI of 0 with 10 is 0.9340805773314359
RSSI of 0 with 11 is 0.9566462058697142
RSSI of 0 with 12 is 0.8558338612524022
RSSI of 0 with 13 is 0.8558338612524022
RSSI of 0 with 14 is 0.8558338612524022
RSSI of 5 with 0 is 0.8973189462481387
RSSI of 5 with 1 is 1.3464559965982086
RSSI of 5 with 2 is 1.00243593354861
RSSI of 5 with 3 is 1.107971614108304
RSSI of 5 with 4 is 1.2961224333697057
RSSI of 5 with 5 is 0.0
RSSI of 5 with 6 is 0.9457807344170609
RSSI of 5 with 7 is 0.6262224446213274
RSSI of 5 with 8 is 1.2938563183172425
RSSI of 5 with 9 is 0.514602684705002
RSSI of 5 with 10 is 1.2947397529224467
RSSI of 5 with 11 is 0.6403440126986368
RSSI of 5 with 12 is 0.8594521111252351
RSSI of 5 with 13 is 0.8594521111252351
RSSI of 5 with 14 is 0.8594521111252351
RSSI of 10 with 0 is 0.9340805773314359
RSSI of 10 with 1 is 1.1994004272029342
RSSI of 10 with 2 is 1.0700747161470219
RSSI of 10 with 3 is 0.8968499601697361
RSSI of 10 with 4 is 1.128890240884116
RSSI of 10 with 5 is 1.2947397529224467
RSSI of 10 with 6 is 0.9990215316213171
RSSI of 10 with 7 is 0.9876997089175108
RSSI of 10 with 8 is 0.7568759544623227
RSSI of 10 with 9 is 1.085232448561597
RSSI of 10 with 10 is 0.0
RSSI of 10 with 11 is 1.0244595661877005
RSSI of 10 with 12 is 1.1150462505325318
RSSI of 10 with 13 is 1.1150462505325318
RSSI of 10 with 14 is 1.1150462505325318

Iteration 3:

Transmission power of 0	3.2102955
Transmission power of 1	3.2102955
Transmission power of 2	3.2102955
Transmission power of 3	3.2102955
Transmission power of 4	3.2102955
Transmission power of 5	3.2102955
Transmission power of 6	3.2102955
Transmission power of 7	3.2102955
Transmission power of 8	3.2102955
Transmission power of 9	3.2102955
Transmission power of 10	3.2102955
Transmission power of 11	3.2102955
Transmission power of 12	3.2102955
Transmission power of 13	3.2102955
Transmission power of 14	3.2102955

Transmission power of 0 = 3.210295501452576	
Transmission power of 1 = 3.210295501452576	
Transmission power of 2 = 3.210295501452576	
Transmission power of 3 = 3.210295501452576	
Transmission power of 4 = 3.210295501452576	
Transmission power of 5 = 3.210295501452576	
Transmission power of 6 = 3.210295501452576	
Transmission power of 7 = 3.210295501452576	
Transmission power of 8 = 3.210295501452576	

```
Transmission power of 9 = 3.210295501452576
Transmission power of 10 = 3.210295501452576
Transmission power of 11 = 3.210295501452576
Transmission power of 12 = 3.210295501452576
Transmission power of 13 = 3.210295501452576
Transmission power of 14 = 3.210295501452576
RSSI of 0 with 0 is 0.0
RSSI of 0 with 1 is 1.0869068342781742
RSSI of 0 with 2 is 0.8636163921551219
RSSI of 0 with 3 is 0.7945841702407067
RSSI of 0 with 4 is 1.0856212226811057
RSSI of 0 with 5 is 0.8077724557930791
RSSI of 0 with 6 is 0.9128577506634165
RSSI of 0 with 7 is 1.006326160057952
RSSI of 0 with 8 is 1.141311991767636
RSSI of 0 with 9 is 1.1259636000354605
RSSI of 0 with 10 is 0.9940252686882967
RSSI of 0 with 11 is 0.9577406810280128
RSSI of 0 with 12 is 0.7895011451502931
RSSI of 0 with 13 is 0.7895011451502931
RSSI of 0 with 14 is 0.7895011451502931
RSSI of 5 with 0 is 0.8077724557930791
RSSI of 5 with 1 is 0.616440302390559
RSSI of 5 with 2 is 0.7066487060456508
RSSI of 5 with 3 is 0.8871470182113448
RSSI of 5 with 4 is 1.313762917215949
RSSI of 5 with 5 is 0.0
RSSI of 5 with 6 is 0.7683709202218325
RSSI of 5 with 7 is 0.8630637334483876
RSSI of 5 with 8 is 0.6598703943751196
RSSI of 5 with 9 is 1.330195083753978
RSSI of 5 with 10 is 1.0700899366511882
RSSI of 5 with 11 is 0.5211412416527612
RSSI of 5 with 12 is 0.9007168927568707
RSSI of 5 with 13 is 0.9007168927568707
RSSI of 5 with 14 is 0.9007168927568707
RSSI of 10 with 0 is 0.9940252686882967
RSSI of 10 with 1 is 0.9939152141934534
RSSI of 10 with 2 is 0.9224973457418948
RSSI of 10 with 3 is 1.2007009983190653
RSSI of 10 with 4 is 1.4188108128322876
```

```
RSSI of 10 with 5 is 1.0700899366511882
RSSI of 10 with 6 is 0.9694632435542233
RSSI of 10 with 7 is 0.9429270250162294
RSSI of 10 with 8 is 0.8253270229072771
RSSI of 10 with 9 is 1.1463557276170186
RSSI of 10 with 10 is 0.0
RSSI of 10 with 11 is 1.0
RSSI of 10 with 12 is 1.2843240781615968
RSSI of 10 with 13 is 1.2843240781615968
RSSI of 10 with 14 is 1.2843240781615968
```

Iteration 4:

Transmission power of 0	3.2102955
Transmission power of 1	3.2102955
Transmission power of 2	3.2102955
Transmission power of 3	3.2102955
Transmission power of 4	3.2102955
Transmission power of 5	3.2102955
Transmission power of 6	3.2102955
Transmission power of 7	3.2102955
Transmission power of 8	3.2102955
Transmission power of 9	3.2102955
Transmission power of 10	3.2102955
Transmission power of 11	3.2102955
Transmission power of 12	3.2102955
Transmission power of 13	3.2102955
Transmission power of 14	3.2102955

Column1	Column2	Column3	Column4
RSSI of 0 with 0 is 0.0			
RSSI of 0 with 1 is 0.8201525365819136			
RSSI of 0 with 2 is 1.1483870253698807			
RSSI of 0 with 3 is 1.2406470965949385			
RSSI of 0 with 4 is 0.858720423264248			
RSSI of 0 with 5 is 1.124437038165089			
RSSI of 0 with 6 is 1.2636878631633552			
RSSI of 0 with 7 is 1.1438882641960377			
RSSI of 0 with 8 is 1.098148137266218			
RSSI of 0 with 9 is 1.0609537620167127			
RSSI of 0 with 10 is 0.9401819144046676			
RSSI of 0 with 11 is 1.1714014390539431			
RSSI of 0 with 12 is 1.0349858701398755			
RSSI of 0 with 13 is 1.0349858701398755			
RSSI of 0 with 14 is 1.0349858701398755			
RSSI of 5 with 0 is 1.124437038165089			
RSSI of 5 with 1 is 1.695229528485765			
RSSI of 5 with 2 is 1.111948873633714			
RSSI of 5 with 3 is 1.1169730273768692			
RSSI of 5 with 4 is 0.8625745142064585			
RSSI of 5 with 5 is 0.0			
RSSI of 5 with 6 is 1.148011095317816			
RSSI of 5 with 7 is 0.8745205769136712			
RSSI of 5 with 8 is 1.1621842424190456			
RSSI of 5 with 9 is 0.7273600062780458			
RSSI of 5 with 10 is 1.3596901866288604			
RSSI of 5 with 11 is 1.1188407125551356			
RSSI of 5 with 12 is 1.567795392735029			
RSSI of 5 with 13 is 1.567795392735029			
RSSI of 5 with 14 is 1.567795392735029			
RSSI of 10 with 0 is 0.9401819144046676			
RSSI of 10 with 1 is 0.9652103142586619			
RSSI of 10 with 2 is 0.9781718598931607			
RSSI of 10 with 3 is 0.6681136288414866			
RSSI of 10 with 4 is 1.4063334615703689			
RSSI of 10 with 5 is 1.3596901866288604			
RSSI of 10 with 6 is 1.0608553960728753			

```
RSSI of 10 with 7 is 1.108258775624581
RSSI of 10 with 8 is 0.7700435985094063
RSSI of 10 with 9 is 1.0799767697473985
RSSI of 10 with 10 is 0.0
RSSI of 10 with 11 is 1.126077189022917
RSSI of 10 with 12 is 0.7794155653752528
RSSI of 10 with 13 is 0.7794155653752528
RSSI of 10 with 14 is 0.7794155653752528
```

Iteration 5:

Transmission power of 0	3.2102955
Transmission power of 1	3.2102955
Transmission power of 2	3.2102955
Transmission power of 3	3.2102955
Transmission power of 4	3.2102955
Transmission power of 5	3.2102955
Transmission power of 6	3.2102955
Transmission power of 7	3.2102955
Transmission power of 8	3.2102955
Transmission power of 9	3.2102955
Transmission power of 10	3.2102955
Transmission power of 11	3.2102955
Transmission power of 12	3.2102955
Transmission power of 13	3.2102955
Transmission power of 14	3.2102955

RSSI of 0 with 0 is 0.0
RSSI of 0 with 1 is 0.9071318465756996
RSSI of 0 with 2 is 1.2435845507910086
RSSI of 0 with 3 is 0.8944027998612734
RSSI of 0 with 4 is 1.0578109340531012
RSSI of 0 with 5 is 1.1607123784320046
RSSI of 0 with 6 is 0.9072913820355786
RSSI of 0 with 7 is 1.0263597610262996
RSSI of 0 with 8 is 1.0688961301708555
RSSI of 0 with 9 is 0.9505070792342064
RSSI of 0 with 10 is 1.0488178601387619
RSSI of 0 with 11 is 0.8955829109986809
RSSI of 0 with 12 is 1.0938146122782297
RSSI of 0 with 13 is 1.0938146122782297
RSSI of 0 with 14 is 1.0938146122782297
RSSI of 5 with 0 is 1.1607123784320046
RSSI of 5 with 1 is 1.6565339738480689
RSSI of 5 with 2 is 0.9656276030321522
RSSI of 5 with 3 is 1.143686990785735
RSSI of 5 with 4 is 1.1495391428318626
RSSI of 5 with 5 is 0.0
RSSI of 5 with 6 is 0.9948722802800064
RSSI of 5 with 7 is 0.8743887068356665
RSSI of 5 with 8 is 1.2895068753412238
RSSI of 5 with 9 is 0.6468064789940776
RSSI of 5 with 10 is 1.2081938867933162
RSSI of 5 with 11 is 1.0333008787491709
RSSI of 5 with 12 is 1.3180414268409217
RSSI of 5 with 13 is 1.3180414268409217
RSSI of 5 with 14 is 1.3180414268409217
RSSI of 10 with 0 is 1.0488178601387619
RSSI of 10 with 1 is 1.1335847348199906
RSSI of 10 with 2 is 0.902276180252048
RSSI of 10 with 3 is 1.0781572575737746
RSSI of 10 with 4 is 1.1456312725353632
RSSI of 10 with 5 is 1.2081938867933162
RSSI of 10 with 6 is 0.9524968816012781
RSSI of 10 with 7 is 1.041217013354371
RSSI of 10 with 8 is 1.0263837571249288
RSSI of 10 with 9 is 0.9794598973535049

```
RSSI of 10 with 10 is 0.0
RSSI of 10 with 11 is 0.810586106270987
RSSI of 10 with 12 is 0.9960272857362734
RSSI of 10 with 13 is 0.9960272857362734
RSSI of 10 with 14 is 0.9960272857362734
```

Analysis:

In every iteration, the RSSI of every observer node with node 12, 13 and 14 are same. Hence, it implies that a Sybil attack has been detected and node 12 is the malicious node.

CHAPTER 5

CONCLUSIONS

5.1 Conclusions

Security is important for many sensor network applications especially if the sensor network protects or monitors critical infrastructures. Security in sensor networks is complicated due to the broadcast nature of the wireless communication. Also, in addition, since the sensor nodes have limited storage and computational resources, Sybil attack proposes a dangerous impact on such a network. A Sybil attack is an attack in which a malicious node in the network illegally claims to have many identities on a single physical device. A Sybil attack can harm the sensor network in more than one way. For example, a Sybil attacker can interrupt location-based or multipath routing, giving the fake impression of being legal nodes on different locations or node-disjoint paths.

In wireless sensor networks, a Sybil attacker can change the aggregated reading outcome by participating many times as a different node. Sybil attack has the ability to disrupt the entire network by taking an inappropriate amount of resources. Sybil node first enters into the cluster and behaves as a node. Then the Sybil node starts impersonating itself and slowly becomes the head of the cluster. This makes the illegitimate node the head of the cluster, thereby disrupting the normal working of the network.

With the help of RSSI based algorithm for the detection of Sybil attack, upon receiving a message, the receiver will associate the RSSI of the message with the sender-id included in the message, and later when another message with same RSSI but with different sender-id is received, the receiver would complain of a Sybil attack. This method of using RSSI for detecting Sybil attack is robust since it detects all Sybil attack cases with 100% completeness and very good accuracy (less than a few percent false positives.) Also since alongside the receiver we need the collaboration of one other node (i.e., only one message communication), this solution is lightweight.

5.2 Future Scope

To find the other types of attacks that can disrupt the Wireless Sensor Networks (WSNs).

We can also work on extending our protocol to tolerate existing Sybil nodes in the network. It would give a deeper insight into Wireless Sensor Network and an even better understanding of Sybil attacks.

Also, since we have already studied how a Sybil attack occurs, it would be great to devise the methods which could defend the network from the Sybil attacks. Some of the existing methods to defend a Sybil attack are Trusted Certification, Resource Testing, Gate Keeper etc. We can implement these techniques to guard against the Sybil attacks and even devise a better method to do the same.

REFERENCES

[1] The Sybil Attack in Sensor Networks: Analysis & Defenses by James Newsome, Elaine Shi, Dawn Song and Adrian Perrig, Carnegie Perrig University.

[2] Wireless Sensor Network: Characteristics and Architectures by Muhammad R Ahman, Xu Hang, Dharmandra Sharma, and Hongyan Sui.

[3] Wireless Sensor Network Applications: A Study in Environment Monitoring System by Mohd Fauzi Othman, Khairunnisa Shazli.

[4] Wikipedia, the free encyclopedia - Information on Micro Aerial Vehicles.

[5] An RSSI based Scheme for Sybil attack detection in Wireless Sensor Networks by Murat Demirbas and Youngwhan Song.

[6] Routing Protocols in Wireless Sensor Networks – A Survey done by Shio Kumar Singh, M P Singh, and D K Singh.

[7] Sybil Attack Type Detection in Wireless Sensor Networks based on Received Signal Strength Indicator detection scheme by Salavat Marian, Popa.

APPENDICES

Source Code:

```
// Using Random Waypoint Model
import java.io.* ;
import java.util.* ;

class Node
{

        int t;// time interval
        Random rand = new Random();
        private int nodeid = 0;
        private static int incr = 0 ;
        double xposition= 0.0; // x coordinate
        double yposition=0.0;  // y coordinate
        double xpositionnew[]=new double[100];
        double ypositionnew[]=new double[100];
        double xpositionprev[]=new double[100];
        double ypositionprev[]= new double[100];

        private double velocity = 0.0 ;

        private double angle ; // movement direction
        private double distanceMatrixNew[][] = new double[100][100];
       // Distance between the nodes
        private double distanceMatrixPrev[][] = new double[100][100];


        private double Ratio[][] = new double[100][100];



        byte edgeMatrix[][] = new byte[100][100] ;
       // Neighbor list of each node


        private double range;

        private double sortedMeanVariance[] = new double[100];
```

```
private int originalPositionArray[]= new  int[100];


private int intArray[] = new int[100];




public Node()
{
        nodeid = incr ;
        xposition = rand.nextInt(600);
        yposition = rand.nextInt(600);
        t = 5;
        velocity = rand.nextInt(11);
        angle = rand.nextInt(360);
        range = 200;
        incr++ ;

}


public void insertSybilDirect(Node N[], int numberofnodes)
{
        int k = 12;

        System.out.print("\n" + k + " is the malicious node\n");

        N[numberofnodes-1].nodeid = numberofnodes-1 ;
        N[numberofnodes-2].nodeid = numberofnodes-2 ;



        N[numberofnodes-1].xposition = N[k].xposition ;
        N[numberofnodes-1].yposition = N[k].yposition ;
        N[numberofnodes-2].xposition = N[k].xposition ;
        N[numberofnodes-2].yposition = N[k].yposition ;
```

```
                N[numberofnodes-1].angle = N[k].angle ;
                N[numberofnodes-2].angle = N[k].angle ;



                N[numberofnodes-1].velocity = N[k].velocity ;
                N[numberofnodes-2].velocity = N[k].velocity ;



        }
        public void currentPosition(Node N[], int numberofnodes)
        {


                double rad = 3.14159/180.0;
                for(int i = 0; i < numberofnodes; i++)
                {

                        if(N[i].angle >=0 && N[i].angle <90)
                        {
                                xpositionnew[i] =  N[i].xposition +
(N[i].velocity*t)*Math.cos(N[i].angle*rad);
                                ypositionnew[i] =  N[i].yposition +
(N[i].velocity*t)*Math.sin(N[i].angle*rad);

                                if(xpositionnew[i] > 600 && ypositionnew[i] < 600)
                                {
                                        xpositionnew[i] = 600 ;
                                        ypositionnew[i] = N[i].yposition +
Math.tan(N[i].angle*rad)*(600-N[i].xposition);
                                        N[i].angle = N[i].angle + 180 ;

                                }
                                if(ypositionnew[i] > 600 && xpositionnew[i] < 600)
                                {
                                        ypositionnew[i] = 600;
                                        xpositionnew[i] = N[i].xposition +
(1.0/Math.tan(N[i].angle*rad))*(600-N[i].yposition);
```

```
                        N[i].angle = N[i].angle + 180;
                    }
                    if(xpositionnew[i] > 600 && ypositionnew[i] > 600)
                    {
                        xpositionnew[i] = 600;
                        ypositionnew[i] = 600;
                        N[i].angle = N[i].angle + 180;
                    }

                }
                if(N[i].angle > 180 && N[i].angle < 270)
                {
                    xpositionnew[i] = N[i].xposition +
(N[i].velocity*t)*Math.cos(N[i].angle*rad);

                    ypositionnew[i] = N[i].yposition
+(N[i].velocity*t)*Math.sin(N[i].angle*rad);

                    if(xpositionnew[i] < 0 && ypositionnew[i] >0)
                    {
                        xpositionnew[i] = 0 ;
                        ypositionnew[i] = N[i].yposition +
Math.tan(N[i].angle*rad)*(0-N[i].xposition);
                        N[i].angle = N[i].angle - 180;
                    }
                    if(ypositionnew[i] <0 && xpositionnew[i] >0)
                    {
                        ypositionnew[i] = 0;
                        xpositionnew[i] = N[i].xposition +
(1.0/Math.tan(N[i].angle*rad))*(0-N[i].yposition);
                        N[i].angle = N[i].angle - 180;
                    }
                    if(xpositionnew[i] < 0 && ypositionnew[i] < 0)
                    {
                        xpositionnew[i] = 0;
                        ypositionnew[i] = 0;
                        N[i].angle = N[i].angle -180;
                    }
                }
```

```
                    if(N[i].angle > 270 && N[i].angle <= 360)
                    {

                            xpositionnew[i] = N[i].xposition +
(N[i].velocity*t)*Math.cos(N[i].angle*rad);

                            ypositionnew[i] = N[i].yposition +
(N[i].velocity*t)*Math.sin(N[i].angle*rad);
                            if(xpositionnew[i] > 600 && ypositionnew[i] > 0)
                            {
                                    xpositionnew[i] = 600 ;
                                    ypositionnew[i] = N[i].yposition +
Math.tan(N[i].angle*rad)*(600-N[i].xposition);
                                    N[i].angle = N[i].angle - 180;
                            }
                            if(ypositionnew[i] < 0 && xpositionnew[i] < 600)
                            {
                                    ypositionnew[i] = 0;
                                    xpositionnew[i] = N[i].xposition +
(1.0/Math.tan(N[i].angle*rad))*(0-N[i].yposition);
                                    angle = angle - 180;
                            }
                            if(xpositionnew[i] > 600 && ypositionnew[i] < 0)
                            {
                                    xpositionnew[i] = 600 ;
                                    ypositionnew[i] = 0;
                                    N[i].angle = N[i].angle - 180;
                            }

                    }

                    if(N[i].angle >90 && N[i].angle <= 180)
                    {
                            xpositionnew[i] = N[i].xposition +
(N[i].velocity*t)*Math.cos(N[i].angle*rad);

                            ypositionnew[i] = N[i].yposition +
(N[i].velocity*t)*Math.sin(N[i].angle*rad);

                            if(xpositionnew[i] < 0 && ypositionnew[i] < 600)
```

```
                {
                        xpositionnew[i] = 0 ;
                        ypositionnew[i] = N[i].yposition +
Math.tan(N[i].angle*rad)*(0-N[i].xposition);
                        N[i].angle = N[i].angle + 180;
                }
                if(ypositionnew[i] > 600 && xpositionnew[i] > 0)
                {
                        ypositionnew[i] = 600;
                        xpositionnew[i] = N[i].xposition +
(1.0/Math.tan(N[i].angle*rad))*(600-N[i].yposition);
                        N[i].angle = N[i].angle + 180;
                }
                if(xpositionnew[i] < 0 && ypositionnew[i] > 600)
                {
                        xpositionnew[i] = 0 ;
                        ypositionnew[i] = 600;
                        N[i].angle = N[i].angle + 180;
                }

        }
        if(N[i].angle ==90)
        {
                xpositionnew[i] = N[i].xposition;
                ypositionnew[i] = N[i].yposition +
(N[i].velocity*t)*Math.sin(N[i].angle*rad);
                if(ypositionnew[i] > 600)
                {
                        ypositionnew[i] = 600;
                        N[i].angle = 270;
                }
        }
        if(N[i].angle ==270)
        {
                xpositionnew[i] = N[i].xposition;
                ypositionnew[i] = N[i].yposition +
(N[i].velocity*t)*Math.sin(N[i].angle*rad);
                if(ypositionnew[i] < 0)
                {
                        ypositionnew[i] = 0;
```

```
                        N[i].angle = 90;
                    }
                }

                xpositionprev[i] = N[i].xposition;
                ypositionprev[i] = N[i].yposition;
                N[i].xposition = xpositionnew[i];
                N[i].yposition = ypositionnew[i];
        }
}

public void updateSpeedDirection(Node N[], int numberofnodes, int k)
{
        for(int i = 0; i < numberofnodes; i++)
        {
                N[i].velocity = rand.nextInt(11);
                N[i].angle = rand.nextInt(360);
        }

        N[numberofnodes-1].xposition = N[k].xposition ;
        N[numberofnodes-1].yposition = N[k].yposition ;
        N[numberofnodes-2].xposition = N[k].xposition ;
        N[numberofnodes-2].yposition = N[k].yposition ;


        N[numberofnodes-1].angle = N[k].angle ;
        N[numberofnodes-2].angle = N[k].angle ;


        N[numberofnodes-1].velocity = N[k].velocity ;
        N[numberofnodes-2].velocity = N[k].velocity ;

}

// Method to compute the mean variance of mobility for each node w.r.t other nodes
public void getDistanceMatrix(int numberofnodes, Node N[])
{
        double p1 =0.0;
        double p2 = 0.0 ;
        double p = 0.0 ;
```

```
double q1 =0.0;
double q2 = 0.0 ;
double q = 0.0 ;
double PathLoss1[] = new double[20];
double PathLoss2[] = new double[20];
double TransmissionPowerDb1[]= new double[100];
double TransmissionPowerMw1[] = new double[100];
double TransmissionPowerDb2[]= new double[100];
double TransmissionPowerMw2[] = new double[100];
double RssiFirst[][]= new double[100][100];
double RssiSecond[][] = new double[100][100];

for(int i = 0; i < numberofnodes; i++)
{
        for(int j = 0; j < numberofnodes; j++)
        {
                p1 = Math.pow( (xpositionprev[i] - xpositionprev[j]), 2);
                p2 = Math.pow( (ypositionprev[i] - ypositionprev[j]), 2);
                p = p1+p2 ;
                distanceMatrixPrev[i][j] = Math.pow( p, 0.5);




                PathLoss1[i] = 20*Math.log10(N[i].range) + 20 *
Math.log10(900) -27.55-2;
                TransmissionPowerDb1[i] = PathLoss1[i] - 85 - 2 + 16.51;
                TransmissionPowerMw1[i] = Math.pow(10,
TransmissionPowerDb1[i]/10.0);
                RssiFirst[i][j] =
TransmissionPowerMw1[i]/(distanceMatrixPrev[i][j]*distanceMatrixPrev[i][j]*1419.78);



                q1 = Math.pow( (xpositionnew[i] - xpositionnew[j]), 2);
                q2 = Math.pow( (ypositionnew[i] - ypositionnew[j]), 2);
                q = q1+q2 ;
                distanceMatrixNew[i][j] = Math.pow( q, 0.5);
```

```
                                PathLoss2[i] = 20*Math.log10(N[i].range) + 20 *
Math.log10(900) -27.55-2;
                                TransmissionPowerDb2[i] = PathLoss2[i] - 85 - 2 + 16.51;
                                TransmissionPowerMw2[i] = Math.pow(10,
TransmissionPowerDb2[i]/10.0);
                                RssiSecond[i][j] =
TransmissionPowerMw2[i]/(distanceMatrixNew[i][j]*distanceMatrixNew[i][j]*1419.78);
                        }
                        System.out.print("\n Transmission power of " + i + " = "  +
TransmissionPowerMw1[i]);
                }



                //Calculate RATIO of Second RSSI and First RSSI
                for (int i = 0; i < numberofnodes; i++ )
                {
                        for (int j = 0; j < numberofnodes; j++ )
                        {
                                if (i != j)
                                {
                                        Ratio[i][j] = RssiSecond[i][j] / RssiFirst[i][j];
                                }
                                else
                                {
                                        Ratio[i][j] = 0 ;
                                }
                        }
                }


        }
        public void displayRatio(int numberofnodes)
        {
                for (int i = 0; i < numberofnodes; i++)
                {
                        for (int j = 0; j < numberofnodes; j++ )
                        {
                                if(i==0 || i== 5 || i == 10)
```

```
                                {

                                        System.out.print("\n RSSI of " + i + " with " +j + " is "
+ Ratio[i][j]);


                                }
                                else
                                        continue;
                        }
                }
        }

        // Compute the edge martix
        public void getEdgeMatrix(Node N[], int numberofnodes)
        {
                int i, j;
                for(i = 0; i < numberofnodes ; i++)
                {
                        for(j = 0; j < numberofnodes; j++)
                        {
                                if(distanceMatrixNew[i][j] <= 200)
                                {
                                        edgeMatrix[i][j] = 1 ;
                                }
                                if(distanceMatrixNew[i][j] > 200)
                                {
                                        edgeMatrix[i][j] = 0;
                                }
                                if(i == j)
                                        edgeMatrix[i][j] = 0 ;
                                if(edgeMatrix[i][j] == 1 && distanceMatrixNew[i][j] >
N[i].range)
                                        edgeMatrix[i][j] = 0;
                                if(edgeMatrix[i][j] == 1 && distanceMatrixNew[i][j] >
N[j].range)
                                        edgeMatrix[i][j] = 0;
                                if(edgeMatrix[i][j] == 1 && distanceMatrixPrev[i][j] >
N[i].range)
                                        edgeMatrix[i][j] = 0;
```

```
                        if(edgeMatrix[i][j] == 1 && distanceMatrixPrev[i][j] >
N[j].range)
                                edgeMatrix[i][j] = 0;

                        if(edgeMatrix[i][j] == 1 && distanceMatrixNew[i][j] >
N[i].range)
                                edgeMatrix[i][j] = 0;
                        if(edgeMatrix[i][j] == 1 && distanceMatrixNew[i][j] >
N[j].range)
                                edgeMatrix[i][j] = 0;
                        if(edgeMatrix[i][j] == 1 && distanceMatrixPrev[i][j] >
N[i].range)
                                edgeMatrix[i][j] = 0;
                        if(edgeMatrix[i][j] == 1 && distanceMatrixPrev[i][j] >
N[j].range)
                                edgeMatrix[i][j] = 0;
                    }
                }
        }
        public void displayEdgeMatrix(int numberofnodes)
        {
                for(int i = 0; i < numberofnodes ; i++)
                {
                        for(int j = 0; j < numberofnodes; j++)
                        {
                                System.out.print("  " + edgeMatrix[i][j]);
                        }
                        System.out.println();
                }
        }
        public void displayDistancePrev(int numberofnodes)
        {
                for(int i = 0; i < numberofnodes; i++)
                {
                        for(int j = 0; j < numberofnodes; j++)
                        {
                                System.out.print("\n The distance of node " + i + " from " + j +
" is: " + distanceMatrixPrev[i][j]);
                        }
                }
```

```
	}
	public void displayDistanceNew(int numberofnodes)
	{
		for(int i = 0; i < numberofnodes; i++)
		{
			for(int j = 0; j < numberofnodes; j++)
			{
				System.out.print("\n The distance of node " + i + " from " + j +
" is: " + distanceMatrixNew[i][j]);
			}
		}
	}
	public void displayNeighbor(int numberofnodes)
	{
		for(int i = 0; i < numberofnodes; i++)
		{
			System.out.print("\n The neighbors of " + i + " = " );
			for(int j = 0;j < numberofnodes; j++)
			{
				if(edgeMatrix[i][j] == 1)
					System.out.print(j + " ");
			}
		}
	}

}

class RSSI2
{
	public static void main(String a[])
	{
		int numberofnodes=15 ;
		//Scanner scan = new Scanner(System.in);
		//System.out.print("\n Enter the number of nodes: ");
		//numberofnodes = scan.nextInt();
		Node N[]= new Node[numberofnodes];
		for(int i = 0; i < numberofnodes ; i++)
			N[i] = new Node() ;
```

```
                Node s[] = new Node[5] ;
                for(int i = 0; i < 5 ; i++)
                {       s[i] = new Node();
                        s[i].insertSybilDirect(N, numberofnodes);
                        s[i].currentPosition(N, numberofnodes);
                        s[i].getDistanceMatrix(numberofnodes, N);

                        //s[i].displayDistancePrev(numberofnodes);
                        //s[i].displayDistanceNew(numberofnodes);

                        s[i].getEdgeMatrix(N, numberofnodes);
                        //s[i].displayDistanceNew(numberofnodes);
                        //s[i].displayNeighbor(numberofnodes);

                        s[i].displayRatio(numberofnodes);
                        System.out.println();

                        s[i].updateSpeedDirection(N, numberofnodes, 12);

                }

        }
}
```